CLAIMS

I claim:

1. A method of specifying, generating, and running a data processing system, comprising:
   (a) providing a plurality of predicates, each referring to a set of arguments,
   (b) providing a plurality of specifications, each of which defines valid argument values for said predicates, and refers to:
      i. a set of variables, and
      ii. a set of clauses,
   (c) specifying some clauses of said specifications by referring to one of said predicates, and to variables of the specification to specify values for arguments of the predicate,
   (d) providing a plurality of plans, which are able to be run to generate sets of valid argument values for said predicates,
   (e) providing a planning means which is able to generate said plans from said specifications,
   (f) providing an evaluation means which is able to run said plans to generate sets of valid argument values,
   (g) generating multiple sets of valid variables for some component steps of some plans, and testing said sets of valid variables for combinations which generate valid result arguments,
   (h) providing an input means which is able to provide input data values for a set of the arguments of a predicate,
   (i) providing an output means which is able to output data,
   (j) inputting argument values from said input means, evaluating input argument values to generate output argument values using said evaluation means, and outputting argument values using said output means.
2. The method of claim 1, further comprising:

(a) specifying some of said clauses by referring to:

i. a set of generating clauses, which will generate a plurality of iterations, each with a set of valid variable values, and

ii. a set of aggregating clauses, which specify a further set of variable values in a current iteration based on the variable values in the previous iteration.

3. The method of claim 2, wherein:

(a) some of said plans refer to an ordering of the clauses of one of said specifications,

(b) said evaluation means further:

i. runs said plans by evaluating each clause in the order specified, using variable values generated by earlier clauses as input to later clauses, and by recursively selecting and running plans for clauses which are specified by predicates,

ii. generates valid variables by backtracking and re-evaluating an antecedent clause to generate further sets of variable values.

4. The method of claim 3, wherein:

(a) said planning means comprises:

i. selecting which clauses of said specification will generate which variable values, by first selecting clauses which generate the minimum number of different values.

5. The method of claim 4, wherein:

(a) said planning means further comprises:

i. ordering the clauses for sequential processing, by first scheduling clauses that have the best combination of processing cost and number of different generated values.

6. The method of claim 5, wherein:

(a) said evaluation means further recursively selects said plans by considering input argument values while running.

7. The method of claim 1, further comprising:

   (a) providing a data store which is able to store and retrieve data,

   (b) specifying some of said clauses by referring to locations in said data store.

8. The method of claim 7:

   (a) further comprising specifying some of said clauses by specifying an alteration to said data store, and

   (b) wherein said evaluation means further comprises:

      i. altering said data store in accordance with the clauses that refer to an alteration in said data store and that are included in a successfully evaluated plan for said input.

9. The method of claim 8, further comprising:

   (a) providing a transaction means of ensuring that alterations to the data store are atomic and independent.

10. The method of claim 9, wherein said data store and the data values for said arguments and said variables are comprised of semi-structured data.

11. The method of claim 7, further comprising:

   (a) providing a plurality of caches which are able to store sets of variable values,

   (b) storing precomputed valid values of some variables of said specifications in said caches,

   (c) providing a plurality of indexes, each of which is able to accept values for a predetermined subset of the variables in one of said caches, and to return all matching sets of the remaining values,

   (d) providing an indexing means which is able to create one of said indexes given one of said caches and a predetermined subset of input variables,

   (e) and wherein said evaluation means further comprises:

      i. generating some variable values by inputing other variable values into said indexes.

12. The method of claim 11:

    (a) further comprising specifying some of said clauses by specifying an alteration to said data store,

    (b) wherein said evaluation means further comprises:

        i. altering said data store in accordance with the clauses that refer to an alteration in said data store and that are included in a successfully evaluated plan for said input.

13. The method of claim 12, wherein:

    (a) some of said plans refer to an ordering of the clauses of one of said specifications,

    (b) said evaluation means further:

        i. runs said plans by evaluating each clause in the order specified, using variable values generated by earlier clauses as input to later clauses, and by recursively selecting and running plans for clauses which are specified by predicates,

        ii. generates valid variables by backtracking and re-evaluating an antecedent clause to generate further sets of variable values, and

        iii. recursively selects said plans by considering input argument values while running.

14. The method of claim 5, further comprising:

    (a) providing a data store which is able to store and retrieve data, and

    (b) specifying some of said clauses by referring to locations in said data store.

15. The method of claim 14:

    (a) further comprising specifying some of said clauses by specifying an alteration to said data store, and

    (b) wherein said evaluation means further comprises:

        i. altering said data store in accordance with the clauses that refer to an alteration in said data store and that are included in a successfully evaluated plan for said input.

16. The method of claim 15, further comprising:

    (a) providing a plurality of caches which are able to store sets of variable values,

    (b) storing precomputed valid values of some variables of said specifications in said caches,

    (c) providing a plurality of indexes, each of which is able to accept values for a predetermined subset of the variables in one of said caches, and to return all matching sets of the remaining values,

    (d) providing an indexing means which is able to create one of said indexes given one of said caches and a predetermined subset of input variables, and

    (e) wherein said evaluation means further comprises:

        i. generating some variable values by inputing other variable values into said indexes.

17. The method of claim 16, further comprising:

    (a) providing a transaction means of ensuring that alterations to the data store are atomic and independent.

18. The method of claim 17, wherein said data store and the data values for said arguments and said variables are comprised of semi-structured data.

19. The method of claim 18, wherein:

    (a) said evaluation means further recursively selects said plans by considering input argument values while running,

    whereby a single declarative language can be used to specify the presentation, business logic, and data layers of a multi-tier application, and

    whereby the same semi-structured, self-describing data model can be used throughout the presentation, business logic, and data layers, and

    whereby the business logic layer specification can be used to drive the automatic creation of the best set of indexes for efficient data retrieval, and

whereby the business logic layer can benefit from the backtracking and search power of rule-based engines, and

whereby the business logic layer can benefit from the optimization efficiency of planning engines normally used only on the data layer, and

whereby the business logic layer can benefit from caching and indexing performance improvements normally reserved for the data layer, and

whereby data layer declarative queries can include user-defined functions, arbitrary iterative algorithms, and object-oriented dynamic-dispatch as part of their specifications, and

whereby applications do not require garbage-collection or any memory management by the programmer, and

whereby application source code can be specified without the run-time side-effects of function arguments passed by reference, of variable assignment and reassignment, or of long-term data store changes, and is thus amenable to automated correctness checking and automated re-writing.